**Spring 2024 – CS Special Topics**

1. CS 494* – Papka – Introduction to High Performance Computing
2. CS 494* – Shweta – AI in Healthcare
3. CS 594 – Chakraborti – Topics in Applied Cryptography
4. CS 594** – Cheng – Socially Responsible Language Models
5. CS 594 – Wang – Linux Kernel Programming

*CS Undergraduate students must submit a modification of major to use the class as a technical elective.

** CS 594 taught by Lu Cheng in Spring 2024 will be converted to CS 517. So, the class will count as a regular CS 5xx coursework and not as a special topics CS 594 for graduation requirements. Students cannot repeat CS 517 in future semesters for credit.
CS 494 – Introduction to High Performance Computing

- Instructor: Michael Papka
- Meeting time: TR 2-3:15pm
- CRN: 42280/42281

Figure 1: Argonne National Laboratory’s newest supercomputer Aurora is currently being installed for operation in 2023/24. Introduction to High-Performance Computing will look at systems like Aurora and answer questions such as what is it made of, how does one program it and how is different from my laptop?

1 Course Description
How are airplanes built without a physical prototype, how do we understand the evolution of the universe, or how are new cancer treatments identified for initial testing? Big problems require big computers - this course is meant to provide a general introduction to the idea of high performance computing and its role in today’s world. This course will discuss what makes up supercomputer, how they are organized and what are the challenges in developing for massive heterogeneous systems.

2 Course Goal
This course aims to introduce students to high-performance computing (HPC) in a general way that is useful to computer science students and all STEM fields. The course will cover fundamental HPC architecture concepts and parallel computing systems software techniques. The course will give students of other domains the needed knowledge to use supercomputers as a vital tool in their quest for new knowledge. The content includes fundamental architecture aspects of shared-memory and distributed-memory systems, as well as paradigms, algorithms, and languages used to program parallel systems. Students will complete several assignments investigating the use of parallel processing systems.

3 Course Work
The course will be lecture based introducing the proposed topics. Additional pointers will be provided to online supplemental material for reinforcement of topics covered. There will be a midterm and final exam, weekly quiz, as well as three or four programming assignments that will implement the concepts presented in class. Access will be provided to needed computational resources to succeed in the course.
4 Tentative Syllabus

4.1 Proposed Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What and Why of High-Performance Computing</td>
</tr>
<tr>
<td>2</td>
<td>Introduction to Class Resources</td>
</tr>
<tr>
<td>3</td>
<td>High Performance Computing Systems</td>
</tr>
<tr>
<td>4</td>
<td>Benchmarking</td>
</tr>
<tr>
<td>5</td>
<td>Resource Management</td>
</tr>
<tr>
<td>6</td>
<td>Multiprocessor and Accelerator Architectures</td>
</tr>
<tr>
<td>7</td>
<td>OpenMP</td>
</tr>
<tr>
<td>8</td>
<td>Message Passing Interface</td>
</tr>
<tr>
<td>9</td>
<td>Midterm Exam</td>
</tr>
<tr>
<td>10</td>
<td>Parallel Algorithms &amp; Libraries</td>
</tr>
<tr>
<td>11</td>
<td>Performance Monitoring</td>
</tr>
<tr>
<td>12</td>
<td>Debugging</td>
</tr>
<tr>
<td>13</td>
<td>Checkpointing</td>
</tr>
<tr>
<td>14</td>
<td>Visualization</td>
</tr>
<tr>
<td>15</td>
<td>Future of High-Performance Computing</td>
</tr>
<tr>
<td>16</td>
<td>Final Exam</td>
</tr>
</tbody>
</table>

4.2 Proposed Evaluation

<table>
<thead>
<tr>
<th>Area</th>
<th>Percent of Final Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Midterm Exam</td>
<td>15%</td>
</tr>
<tr>
<td>Final Exam</td>
<td>15%</td>
</tr>
<tr>
<td>Assignments</td>
<td>60%</td>
</tr>
<tr>
<td>Quiz &amp; Participation</td>
<td>10%</td>
</tr>
</tbody>
</table>

4.3 Book and Readings

Course material will be drawn from the following sources (preliminary list):

- Miscellaneous readings from The International Conference for High Performance Computing, Networking, Storage, and Analysis, International Conference on Supercomputing, and International Parallel and Distributed Processing Symposium as well as others.


All material needed for course will be available via lecture or online sources. High Performance Computing Modern Systems and Practices is recommended but not required for the course.

4.4 Prerequisite

Students are expected to have taken and received an 'C' or better in CS 251 Data Structures and comfortable with programming in C/C++.
CS 494 – AI in Healthcare
- Instructor: Shweta Yadav
- Meeting time: MW 9:30-10:45am
- CRN: 42278/42289

Course Description
In the recent past, there has been a dramatic revolution in Artificial Intelligence (AI). One domain where AI has made significant breakthroughs is in healthcare, demonstrating its application from drug discovery to precision medicine. This course is designed to provide students with an in-depth understanding of recent advancements in AI in healthcare. We will begin with fundamental concepts of deep learning, natural language processing, and computer vision in the context of healthcare. Thereafter, we will explore various interesting applications of AI in healthcare and thoroughly examine AI models tailored to process various forms of healthcare data, such as electronic medical records, medical images, social media data, and multi-omics data. In the latter part of the course, we will cover advanced topics, including ethics, fairness, interpretability, and robustness in healthcare AI.

Course Work
- Weekly reading
- Three programming assignments to gain expertise in using deep learning methods with healthcare data.
- Semester-long group projects to develop and implement an AI-based approach to a healthcare problem. Each team of three students must submit a project, which includes a proposal submission, a project presentation (once during week seven and again during week fifteen), and a final report submission in the format of an AI conference paper.
- Final exam.

Textbook and Readings

Prerequisites
CS 411. Artificial Intelligence I
Grading Policy

- Project (proposal, presentations, and final report 70%)
- Final Exam (30%)

Tentative Course Outline

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td>Introduction to AI in Healthcare: background, challenges, applications</td>
</tr>
<tr>
<td>Week 2</td>
<td>Introduction to Types of Healthcare Data</td>
</tr>
<tr>
<td>Week 3</td>
<td>Fundamentals of Natural Language Processing for Healthcare</td>
</tr>
<tr>
<td>Week 4</td>
<td>Fundamentals of Deep Learning for Healthcare</td>
</tr>
<tr>
<td>Week 5</td>
<td>Multimodal Data, Multimodal Models, Weakly and Self-Supervised Learning</td>
</tr>
<tr>
<td>Week 6</td>
<td>Medical Images: Classification, Detection, and Segmentation</td>
</tr>
<tr>
<td>Week 7</td>
<td>Project Proposal Presentation</td>
</tr>
<tr>
<td>Week 8</td>
<td>Drug Discovery and Precision Medicine</td>
</tr>
<tr>
<td>Week 9</td>
<td>Disease Diagnosis and Treatment Recommendations from Electronic Health Records</td>
</tr>
<tr>
<td>Week 10</td>
<td>Introduction to Multi-omics</td>
</tr>
<tr>
<td>Week 11</td>
<td>Generative AI in Healthcare</td>
</tr>
<tr>
<td>Week 12</td>
<td>The Role of Social Media in the Healthcare Domain</td>
</tr>
<tr>
<td>Week 13</td>
<td>Interpretability, Fairness, and Ethics</td>
</tr>
<tr>
<td>Week 14</td>
<td>Distributed Learning, Security and Privacy</td>
</tr>
<tr>
<td>Week 15</td>
<td>Final Project Presentation</td>
</tr>
<tr>
<td>Week 16</td>
<td>Final Exam</td>
</tr>
</tbody>
</table>
Spring 2024 – CS Special Topics

CS 594 – Topics in Applied Cryptography
- Instructor: Anrin Chakraborti
- Meeting time: TR 2:00-3:15 pm
- CRN: 33792

Course Description

In a complex and interconnected computing ecosystem, security and privacy of both data and user identities are of key importance. In many real world systems, these assurances are provided using cryptographic tools. Cryptography in many ways drives the economy by making digital content secure. In addition, it assigns authenticated digital identities, and facilitates transactions without interference from bad actors. To facilitate these needs, the field of cryptography has advanced rapidly beyond building data encryption protocols that provide communication secrecy, to a significantly richer (and more complex) tool set.

However, studying cryptographic tools in isolation is not enough to understand how they are deployed in practice. This course will take an application-oriented approach and introduce students to systems where cryptographic tools are used today. On the theoretical end, the content will cover descriptions of advanced cryptographic primitives like secure multiparty computation and zero-knowledge proofs. To demonstrate applications of these tools, the content will cover end-to-end encrypted messaging services, e-cash technologies (cryptocurrencies), encrypted database designs and applications of cryptographic tools to machine learning and e-voting schemes. Students will complete a project on a topic of their choice demonstrating a real application of a cryptographic tool.

Course Outline

The course will be (logically) divided into two segments. The segment on fundamentals will introduce commonly used cryptographic tools. The segment on applications will demonstrate how these cryptographic tools are used in real-world systems. The required mathematical background will be provided.

Fundamentals

- Week 1: Introduction + mathematical tools required for this course
- Week 2: Private key encryption schemes: definitions, constructions
- Week 3: Public key encryptions schemes: definitions, constructions
- Week 4: Hash functions, digital signature schemes
**Spring 2024 – CS Special Topics**

- Week 5: Advanced encryption schemes: homomorphic encryption, lattice-based cryptography, pairing-based cryptography
- Week 6: Secure multiparty computation: fundamentals & constructions
- Week 7: Zero-knowledge proofs & applications

**Applications**

- Week 8: End-to-end encrypted messaging services
- Week 9: Anonymous communication: The dining cryptographer’s problem, mixnets, crowds
- Week 10: E-cash systems: Chaum’s digital cash and modern cryptocurrencies
- Week 11: Encrypted databases and searchable encryption
- Week 12: Side channels in cryptographic systems
- Week 13: Cryptographic tools in ML: verifiability and privacy
- Week 14: Applications of non-interactive zero-knowledge proofs to content moderation, blockchains and e-voting
- Week 15: Project demo and presentations

**Coursework**

The course work will include lectures based on the reading material and student presentations. Students will complete a course project on a topic of their choice.

**Recommended reading**

There is no required textbook for the course. However, the material covered will be from the following sources

- A Graduate Course in Applied Cryptography by Boneh & Shoup (Version 0.6)
- Cryptography Engineering: Design Principles and Practical Applications by Ferguson, Schneier, Kohno (1st edition)
- Research papers from recent security and cryptography conferences

**Prerequisite**

Grade of C or better in CS 251; and IE 342 or STAT 381 or STAT 401 or CS 488
Spring 2024 – CS Special Topics

Grading Criteria

This course only has two components. This includes a research presentation that will require students to present a survey of recent papers on a topic that is covered in class, and a final project on a topic of their choice. The course will not include assignments, a midterm or a final.

- In-class research presentation: 30%
- Final project: 70%
CS 594 – Socially Responsible Language Models

- Instructor: Lu Cheng
- Meeting time: TR 8:00-9:15am
- CRN: 33648

Course Description

Language models (LMs) have become prevalent in everyday life in the form of personal assistants, chatbots, writing, and summarizing tools with improved efficiency and accuracy of natural language processing (NLP). Large Language models (LLM)-based chatbots such as ChatGPT and Llama-2 have shown successfully passed the United States Medical Licensing Exam (USMLE), Radiology Board-style Examination, Law exam, etc. Therefore, an era where LMs play a significant role in our day-to-day life is not far away. LLMs carry significant implications across various industries and knowledge-based applications. However, the adoption of LLMs also raises ethical and social concerns. Some of these include the lack of interpretability in language models, potential biases and discrimination in the generated content, privacy leakage, model vulnerability, dissemination of fake and misleading content, copyright and plagiarism concerns, and the environmental impact associated with training and using LLM models. In light of these risks, it is imperative to develop and implement LLM models and applications in accordance with responsible AI principles. The course will focus on both the theoretical and practical challenges related to the design and deployment of responsible LLMs and will have strong multidisciplinary components, including contributions about policy, legal issues, and the societal impact of responsible LLMs.

Course work

1. Reading, discussion, presentation of research papers (40%)
2. Two course projects (60%)

Note 1: Each student will be asked to present from 2 papers, and to be the discussant for 2 other papers – this means writing a short-written critique for the paper in question and be ready to participate in discussion. Exact workload will depend on the class size.

Note 2: Attendance is required. The instructor will start taking attendance after the add/drop period.

Prerequisite

Students are required to have taken and received at least a ‘C’ in one of the following courses: 412 Introduction to Machine Learning, 421 Natural Language Processing, 521 statistical NLP, 533 Deep learning for NLP, 583 Data Mining and Text Mining, 559 Neural Networks.
## Course Outline (Tentative)

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td><strong>Weeks 1-9: Theories</strong></td>
</tr>
<tr>
<td>Week 2</td>
<td>Background on Language</td>
</tr>
<tr>
<td>Week 3</td>
<td>Models Bias and (un)Fairness</td>
</tr>
<tr>
<td>Week 4</td>
<td>Uncertainty quantification</td>
</tr>
<tr>
<td>Week 5</td>
<td>Transparency</td>
</tr>
<tr>
<td>Weeks 6-7</td>
<td>Privacy and copyright</td>
</tr>
<tr>
<td>Week 8</td>
<td>implications Other societal</td>
</tr>
<tr>
<td>Week 9</td>
<td>Project I presentations</td>
</tr>
<tr>
<td>Week 10-14</td>
<td><strong>Weeks 10-14: Paper presentations and discussions</strong></td>
</tr>
<tr>
<td>Week 15</td>
<td>Project II Presentations</td>
</tr>
</tbody>
</table>
CS 594 – Linux Kernel Programming

- Instructor: Xiaoguang Wang
- Meeting time: TR 9:30-10:45am
- CRN: 34724

**Course Description**

The Linux kernel is one of the most commonly used (and heavily optimized) operating system kernels with wide acceptance in the industry. It is used on a broad spectrum of computer hardware, from embedded devices to servers, and from portable devices to HPC platforms. Due to its diverse properties, several industrial systems are based on Linux (e.g., Google Android), and many academic systems software research projects are conducted on Linux. Linux kernel skills are highly useful for software engineers, especially those involved with systems software, but also for hardware engineers to test new features or devices.

Similar courses of operating systems teach smaller kernels or focus on only a few subsystems of the Linux kernel. This is not sufficient for students to obtain a deeper understanding of Linux and be able to implement non-trivial components (e.g., new virtual memory managers, file systems, schedulers, and eBPF subsystems). Moreover, courses based on Unix often teach how to interact with the kernel but not the mechanisms implemented within it. Students exposed to this course will learn how to program the Linux kernel, implement new or modify existing kernel subsystems, and performance-optimize kernel modules and subsystems by exploiting various time/space tradeoffs and building experience working with a large-scale open-source project.

In addition, students will learn the differences between designing, implementing, and debugging application-level and system-level software. These skills are highly desirable for developing operating systems, embedded systems, virtualization infrastructures, and even application software development.

**Prerequisites**

Systems programming (e.g., CS 361), or operating systems (e.g., CS 461) or equivalent, or consent of the instructor
Programming Requirements

Good knowledge of C programming and the Linux command line is assumed. Knowledge concerning algorithms, data structures, and computer architecture is recommended.

Course Outline and Topics (tentative)

The course will combine lectures on Linux Kernel programming and reading discussions. Students are encouraged to read papers on the state-of-the-art of systems research.

<table>
<thead>
<tr>
<th>Topics (tentative)</th>
<th>Req. Book Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalities about Linux and software engineering techniques for large projects (version control, toolchains, configure, make, kernel installation, kernel code exploration/browsing)</td>
<td>1, 2, 5, 20</td>
</tr>
<tr>
<td>Kernel debugging techniques (printk, kernel traces/trace, kernel space debugging/gdb, QEMU/KVM debugging, gdb scripting)</td>
<td>18</td>
</tr>
<tr>
<td>Device drivers and data structures (generic device drivers, kernel basic data structures, kernel objects)</td>
<td>6, 17</td>
</tr>
<tr>
<td>Memory management (segmentation, paging, software MMU, physical memory space and allocation schemes, get_free_page / kmalloc / vmalloc / cache_alloc and NUMA affinity, process address space and vma_struct, mm_struct, remapping functions, I/O mapping)</td>
<td>12, 15</td>
</tr>
<tr>
<td>Linux process model and scheduling (kernel / user process context and task descriptor, kthreads, nptl, process and lightweight threading models, syscalls, kernel / libc / application interaction, scheduling and context switching)</td>
<td>3, 4</td>
</tr>
<tr>
<td>Kernel synchronization and time management</td>
<td>9, 10, 11</td>
</tr>
<tr>
<td>Interrupt / exception handling (interrupt paths, hardware and software interrupts, synchronous exception handling, signals, system call fast path)</td>
<td>7, 8</td>
</tr>
<tr>
<td>Virtual file system (block layer and block cache, file system driver, I/O scheduler, extended file system)</td>
<td>13, 14, 16</td>
</tr>
</tbody>
</table>

Course Work

The course will have 3-4 small projects and one final project (no final exam). The final project will be a mini research style project, including a research study proposal, a prototype with design and implementation, and a prototype evaluation. A final project presentation will be scheduled on the last day of the instruction session.